第3章　可视化TensorFlow

TensorFlow技术解析与实战 (/book/details/4862)

可视化是认识程序的最直观方式。在做数据分析时，可视化一般是数据分析最后一步的结果呈现。把可视化放到“基础篇”，是为了让读者在安装完成后　就能先看一下TensorFlow到底有哪些功能，直观感受一下深度学习的学习成果，让学习目标一目了然。

3.1　PlayGround

PlayGround[1]是一个用于教学目的的简单神经网络的在线演示、实验的图形化平台，非常强大地可视化了神经网络的训练过程。使用它可以在浏览器里训练神经网络，对Tensorflow有一个感性的认识。

PlayGround界面从左到右由数据（DATA）、特征（FEATURES）、神经网络的隐藏层（HIDDEN LAYERS）和层中的连接线和输出（OUTPUT）几个部分组成，如图3-1所示。

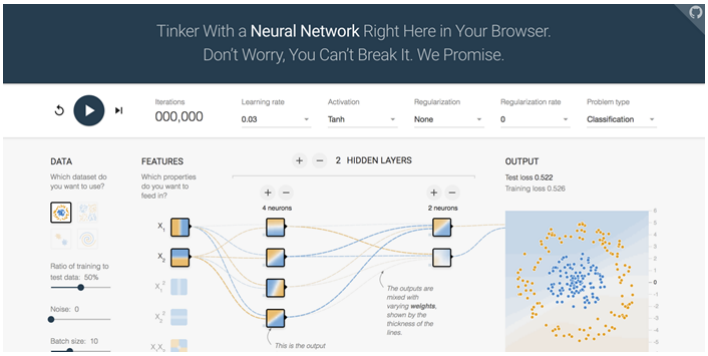

图3-1

3.1.1　数据

在二维平面内，点被标记成两种颜色。深色（电脑屏幕显示为蓝色）代表正值，浅色（电脑屏幕显示为黄色）代表负值。这两种颜色表示想要区分的两类，如图3-2所示。

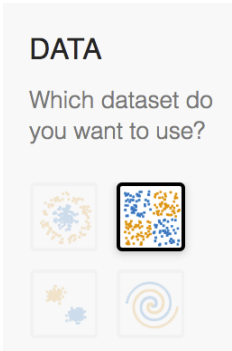

图3-2

网站提供了4种不同形态的数据，分别是圆形、异或、高斯和螺旋，如图3-3所示。神经网络会根据所给的数据进行训练，再分类规律相同的点。

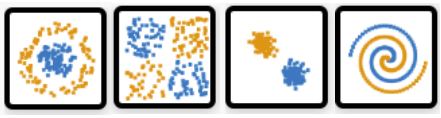

图3-3

PlayGournd中的数据配置非常灵活，可以调整**噪声**（noise）的大小。图3-4展示的是噪声为0、25和50时的数据分布。

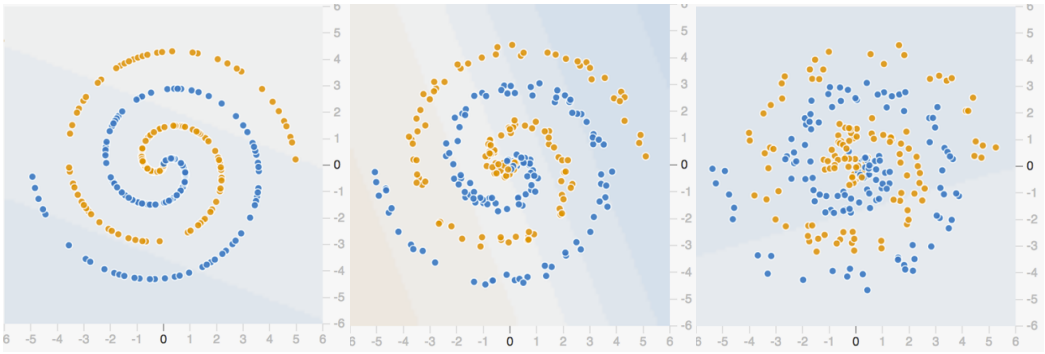

图3-4

PlayGournd中也可以改变训练数据和测试数据的比例（ratio）。图3-5展示的是训练数据和测试数据比例为1：9和9：1时的情

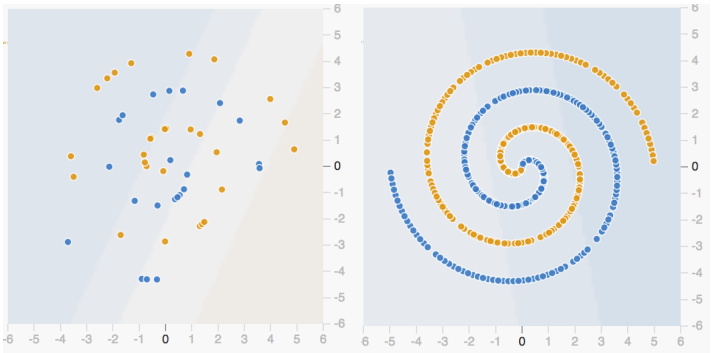

图3-5

此外，PlayGournd中还可以调整输入的每批（batch）数据的多少，调整范围可以是1～30，就是说每批进入神经网络数据的点可以1～30个，如图3-6所示。

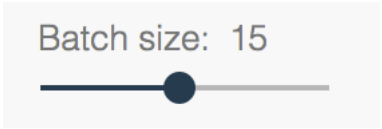

图3-6

3.1.2 特征

接下来我们需要做**特征提取**（feature extraction），每一个点都有X_1和X_2两个特征，由这两个特征还可以衍生出许多其他特征，如X_1X_1、X_2X_2、X_1X_2、$\sin(X_1)$、$\sin(X_2)$等，如图3-7所示。

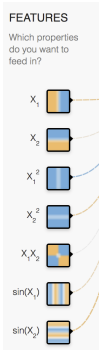

图3-7

从颜色上，X_1左边浅色（电脑屏幕显示为黄色）是负，右边深色（电脑屏幕显示为蓝色）是正，X_1表示此点的横坐标值。同理，X_2上边深色是正，下边浅色是负，X_2表示此点的纵坐标值。X_1X_1是关于横坐标的“抛物线”信息，X_2X_2是关于纵坐标的“抛物线”信息，X_1X_2是“双曲抛物面”的信息，$\sin(X_1)$是关于横坐标的“正弦函数”信息，$\sin(X_2)$是关于纵坐标的“正弦函数”信息。

因此，我们要学习的**分类器**（classifier）就是要结合上述一种或者多种特征，画出一条或者多条线，把原始的蓝色和黄色数据分开。

3.1.3 隐藏层

我们可以设置隐藏层的多少，以及每个隐藏层神经元的数量，如图3-8所示。

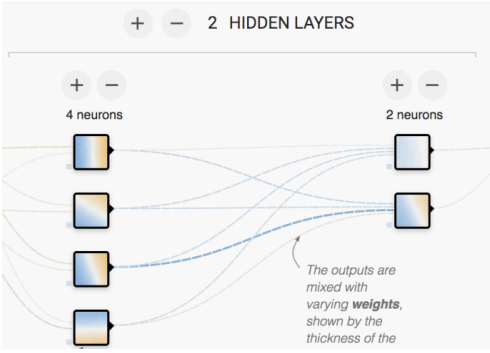

图3-8

隐藏层之间的连接线表示**权重**（weight），深色（蓝色）表示用神经元的原始输出，浅色（黄色）表示用神经元的负输出。连接线的粗细和深浅表示权重的绝对值大小鼠标放在线上可以看到具体值，也可以修改值，如图3-9所示。

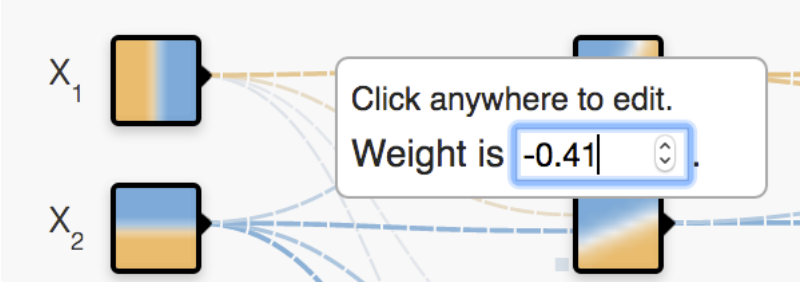

图3-9

修改值时，同时要考虑激活函数，例如，当换成Sigmoid时，会发现没有负向的黄色区域了，因为Sigmoid的值域是(0,1)，如图3-10所示。自然语言处理

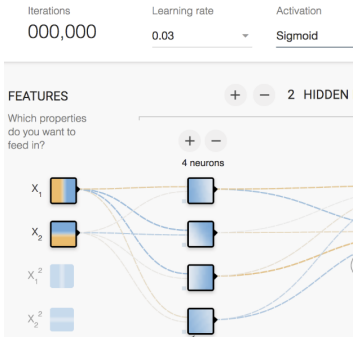

图3-10

下一层神经网络的神经元会对这一层的输出再进行组合。组合时，根据上一次预测的准确性，我们会通过反向传播给每个组合不同的权重。组合时连接线的粗细和深浅会发生变化，连接线的颜色越深越粗，表示权重越大。

3.1.4 输出

输出的目的是使黄色点都归于黄色背景，蓝色点都归于蓝色背景，背景颜色的深浅代表可能性的强弱。

我们选定螺旋形数据，7个特征全部输入，进行试验。选择只有3个隐藏层时，第一个隐藏层设置8个神经元，第二个隐藏层设置4个神经元，第三个隐藏层设置2个神经元。训练大概2分钟，测试损失（test loss）和训练损失（training loss）就不再下降了。训练完成时可以看出，我们的神经网络已经完美地分离出了橙色点和蓝色点，如图3-11所示。

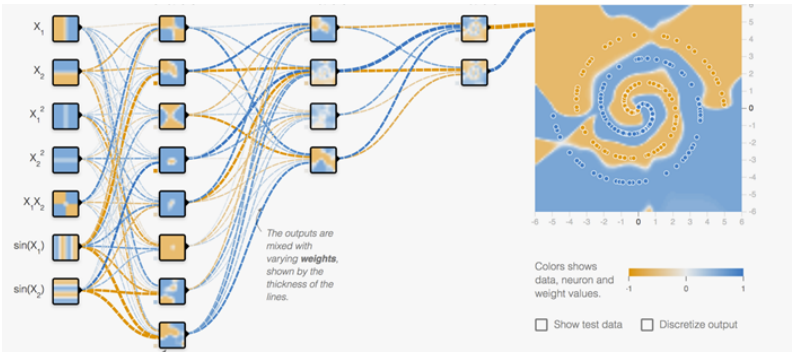

图3-11

假设我们只输入最基本的前4个特征，给足多个隐藏层，看看神经网络的表现。假设加入6个隐藏层，前4层每层有8个神经元，第五层有6个神经元，第六层有2个神经元。结果如图3-12所示。

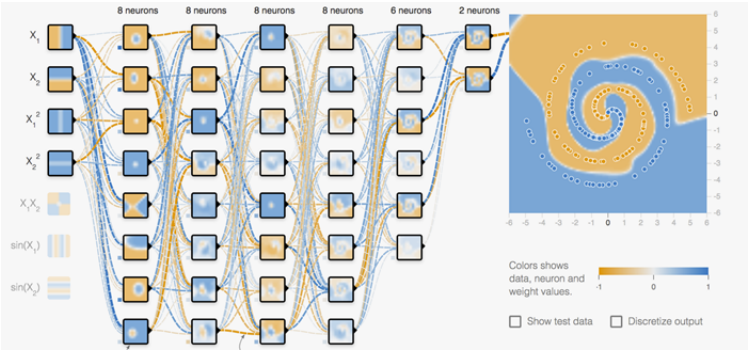

图3-12

我们发现，通过增加神经元的个数和神经网络的隐藏层数，即使没有输入许多特征，神经网络也能正确地分类。但是，假如我们要分类的物体是猫猫狗狗的图片，而不是肉眼能够直接识别出特征的黄点和蓝点呢？这时候怎样去提取那些真正有效的特征呢？

有了神经网络，我们的系统自己就能学习到哪些特征是有效的、哪些是无效的，通过自己学习的这些特征，就可以做到自己分类　这就大大提高了我们解决语音　图像这种复杂抽象问题的能力。

3.2　TensorBoard[2]

TensorBoard是TensorFlow自带的一个强大的可视化工具，也是一个Web应用程序套件。TensorBoard目前支持7种可视化　即SCALARS　IMAGES　AUDIO　GRAPHS、DISTRIBUTIONS、HISTOGRAMS和EMBEDDINGS。这7种可视化的主要功能如下。

- SCALARS：展示训练过程中的准确率、损失值、权重/偏置的变化情况。
- IMAGES：展示训练过程中记录的图像。
- AUDIO：展示训练过程中记录的音频。
- GRAPHS：展示模型的数据流图，以及训练在各个设备上消耗的内存和时间。
- DISTRIBUTIONS：展示训练过程中记录的数据的分布图。
- HISTOGRAMS：展示训练过程中记录的数据的柱状图。
- EMBEDDINGS：展示词向量（如Word2vec）后的投影分布。

TensorBoard通过运行一个本地服务器，来监听6006端口。在浏览器发出请求时，分析训练时记录的数据，绘制训练过程中的图像。在9.3节的MNIST示例中，会逐一讲解TensorBoard的图像绘制，让读者更好地了解训练的过程中发生了什么。本节我们就先看一下TensorBoard能够绘制出哪些东西。

TensorBoard的可视化界面如图3-13所示。

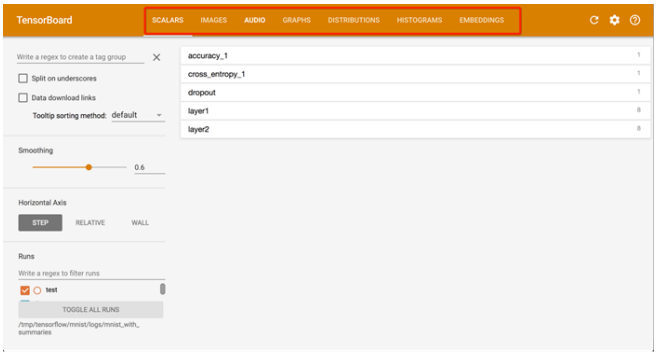

图3-13

从图3-13中可以看到，在标题处有上述几个可视化面板，下面通过一个示例，分别介绍这些可视化面板的功能。

这里，我们运行手写数字识别的入门例子，如下：

```
python tensorflow-1.1.0/tensorflow/examples/tutorials/mnist/mnist_with_summaries.py
```

然后，打开TensorBoard面板：

```
tensorboard --logdir=/tmp/tensorflow/mnist/logs/mnist_with_summaries
```

这时，输出：

```
Starting TensorBoard 39 on port 6006
(You can navigate to http://192.168.0.101:6006)
```

我们就可以在浏览器中打开http://192.168.0.101:6006，查看面板的各项功能。

3.2.1 SCALARS面板

SCALARS面板的左边是一些选项，包括Split on undercores（用下划线分开显示）、Data downloadlinks（数据下载链接）、Smoothing（图像的曲线平滑程度）以及Horizontal Axis（水平轴）的表示，其中水平轴的表示分3种（STEP代表迭代次数，RELATIVE代表按照训练集和测试集的相对值，WALL代表按照时间），如图3-14左边所示。图3-14右边给出了准确率和交叉熵损失函数值的变化曲线（迭代次数是1000次）。

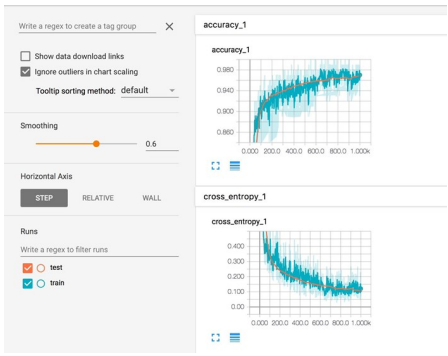

图3-14

SCALARS面板中还绘制了每一层的**偏置**（biases）和**权重**（weights）的变化曲线，包括每次迭代中的最大值、最小值、平均值和标准差　如图3 15所示

图3-15

3.2.2 IMAGES面板

图3-16展示了训练数据集和测试数据集经过预处理后图片的样子。

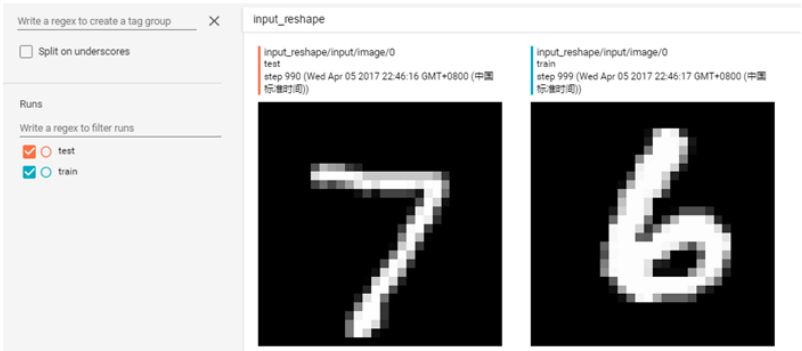

图3-16

3.2.3 AUDIO面板

AUDIO面板是展示训练过程中处理的音频数据。这里暂时没有找到合适的例子，读者了解即可。

3.2.4 GRAPHS面板

GRAPHS面板是对理解神经网络结构最有帮助的一个面板，它直观地展示了数据流图。图 3-17 所示界面中节点之间的连线即为数据流，连线越粗，说明在两个节点之间流动的张量（tensor）越多。

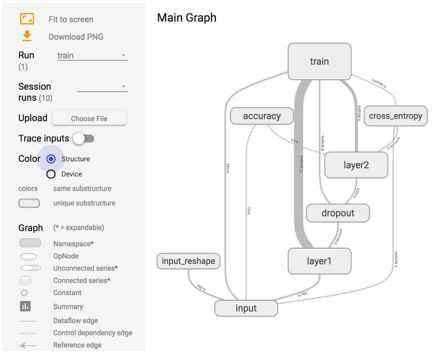

图3-17

在GRAPHS面板的左侧，可以选择迭代步骤。可以用不同Color（颜色）来表示不同的Structure（整个数据流图的结构），或者用不同Color来表示不同Device（设备）。例如，当使用多个GPU时，各个节点分别使用的GPU不同。

当我们选择特定的某次迭代（如第899次）时，可以显示出各个节点的Compute time（计算时间）以及Memory（内存消耗），如图3-18所示

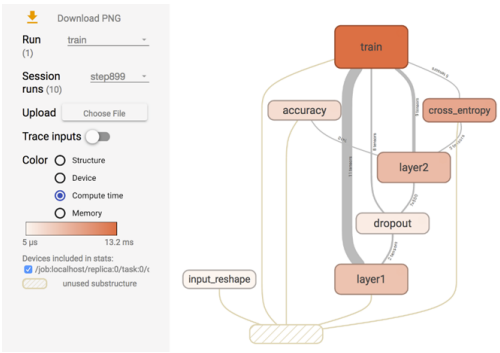

图3-18

3.2.5 DISTRIBUTIONS面板

DISTRIBUTIONS面板和接下来要讲的HISTOGRAMS面板类似，只不过是用平面来表示来自特定层的激活前后、权重和偏置的分布。图3-19展示的是激活之前和激活之后的数据分布。

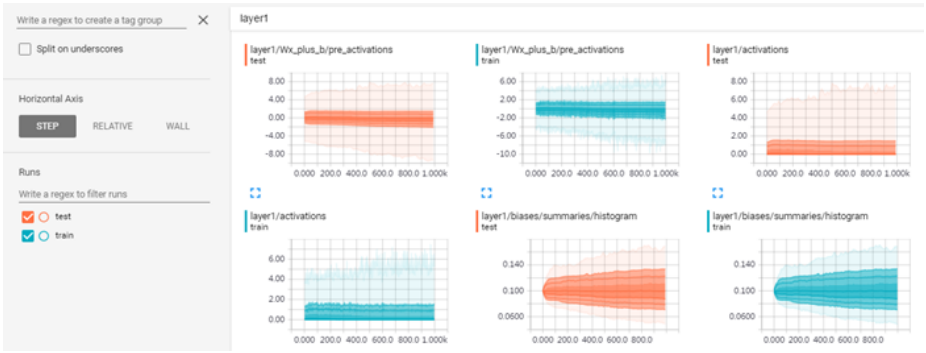

图3-19

3.2.6 HISTOGRAMS面板

HISTOGRAMS主要是立体地展现来自特定层的激活前后、权重和偏置的分布。图3-20展示的是激活之前和激活之后的数据分布。

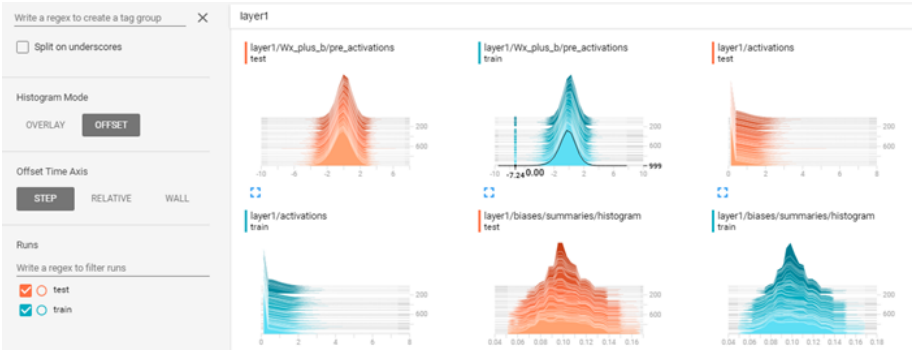

图3-20

3.2.7 EMBEDDINGS面板

EMBEDDINGS面板在MNIST这个示例中无法展示，在3.3节中我们会用Word2vec例子来看一下这个面板的词嵌入投影仪。

3.3 可视化的例子

词嵌入（word embedding）在机器学习中非常常见，可以应用在自然语言处理、推荐系统等其他程序中。下面我们就以Word2vec为例来看看词嵌入投影仪的可视化。

TensorFlow的Word2Vec有basic、optimised这两个版本，我们重点来看这两个版本的可视化表示。

3.3.1 降维分析

本节将以GitHub上的一段代码[3]为例，讲述可视化的思路。

Word2vec采用text8[4]作为文本的训练数据集。这个文本中只包含a～z字符和空格，共27种字符。我们重点讲述产生的结果可视化的样子以及构建可视化的过程。这里我们采用的是Skip-gram模型，即根据目标词汇预测上下文。也就是说，给定n个词围绕着词w，用w来预测一个句子中其中一个缺漏的词　以概率p(c|w)来表示。最后生成的用t-SNE降维呈现词汇接近程度的关系如图3-21所示。

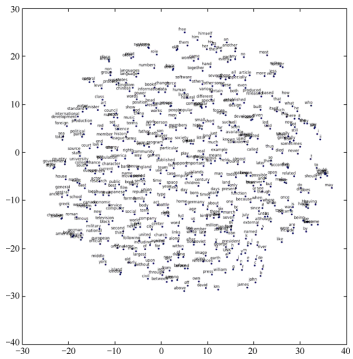

图3-21

在word2vec_basic.py中，从获得数据到最终得到可视化的结果的过程分为5步。

（1）下载文件并读取数据。主要是read_data函数，它读取输入的数据，输出一个list，里面的每一项就是一个词。

```
def read_data(filename):
  with zipfile.ZipFile(filename) as f:
    data = tf.compat.as_str(f.read(f.namelist()[0])).split()
  return data
```

这里的data就类似于['fawn', 'homomorphism', 'nordisk', 'nunnery']。

（2）建立一个词汇字典。这里首先建立了一个词汇字典，字典里是对应的词和这个词的编码。

```
vocabulary_size = 50000

def build_dataset(words):
  count = [['UNK', -1]]
  count.extend(collections.Counter(words).most_common(vocabulary_size - 1))
  dictionary = dict()
  for word, _ in count:
    dictionary[word] = len(dictionary)
  data = list()
  unk_count = 0
  for word in words:
    if word in dictionary:
      index = dictionary[word]
    else:
      index = 0  # dictionary['UNK']
      unk_count += 1
    data.append(index)
  count[0][1] = unk_count
  reverse_dictionary = dict(zip(dictionary.values(), dictionary.keys()))
  return data, count, dictionary, reverse_dictionary

data, count, dictionary, reverse_dictionary = build_dataset(words)
```

dictionary里存储的就是词与这个词的编码；reverse_dictionary是反过来的dictionary，对应的是词的编码与这个词；data是list，存储的是词对应的编码，也就是第一步中得到的词的list，转化为词的编码表示；count中存储的是词汇和词频，其中重复数量少于49 999个词，用'UNK'来代表稀有词。具体示例如下：

```
data [5239, 3084, 12, 6, 195, 2, 3137, 46, 59, 156]
count [['UNK', 418391], ('the', 1061396), ('of', 593677), ('and', 416629),
       ('one', 411764), ('in', 372201), ('a', 325873), ('to', 316376), ('zero', 264975),
           ('nine', 250430)]
dictionary {'fawn': 0, 'homomorphism': 1, 'nordisk': 2, 'nunnery': 3, 'chthonic':
             4, 'sowell': 5, 'sonja': 6, 'showa': 7, 'woods': 8, 'hsv': 9}
reverse_dictionary {0: 'fawn', 1: 'homomorphism', 2: 'nordisk', 3: 'nunnery', 4:
                      'chthonic', 5: 'sowell', 6: 'sonja', 7: 'showa', 8: 'woods', 9: 'hsv'}
```

（3）产生一个批次（batch）的训练数据。这里定义generate_batch函数，输入batch_size、num_skips和skip_windows，其中batch_size是每个batch的大小，num_skips代表样本的源端要考虑几次，skip_windows代表左右各考虑多少个词，其中skip_windows*2=num_skips。最后返回的是batch和label，batch的形状是[batch_size]，label的形状是[batch_size, 1]，也就是用一个中心词来预测一个周边词。

举个例子。假设我们的句子是“我在写一首歌”，我们将每一个字用dictionary中的编码代替，就变成了[123, 3084, 12, 6, 195, 90]，(假设这里的window_size是3)也就是只预测上文一个词，下文一个词，假设我们的generate_batch函数从3084出发，源端重复2次，那么batch就是[3084 3084 12 12 6 6 195 195]，3084的上文是123，下文是12；12的上文是3084，下文是6；6的上文是12，下文是195；195的上文是6，下文是90。因此，对应输出的label就是：

```
[[ 123]
 [12]
 [3084]
 [    6]
 [  12]
 [ 195]
 [   6]
 [   90]]
```

（4）构建和训练模型。这里我们构建一个Skip-gram模型，具体模型搭建可以参考Skip-gram的相关论文。执行结果如下：

```
Found and verified text8.zip
Data size 17005207 # 共有17005207个单词数
Most common words (+UNK) [['UNK', 418391], ('the', 1061396), ('of', 593677),
                          ('and', 416629), ('one', 411764)]
Sample data [5239, 3084, 12, 6, 195, 2, 3137, 46, 59, 156] ['anarchism', 'originated',
           'as', 'a', 'term', 'of', 'abuse', 'first', 'used', 'against']
3084 originated -> 5239 anarchism
3084 originated -> 12 as
12 as -> 3084 originated
12 as -> 6 a
6 a -> 195 term
6 a -> 12 as
195 term -> 6 a
195 term -> 2 of
Initialized
Average loss at step  0 :  263.743347168
Nearest to a: following, infantile, professor, airplane, retreat, implicated,
ideological, epstein,
Nearest to will: apokryphen, intercity, casta, nsc, commissioners, conjuring,
stockholders, bureaucrats,
Nearest to this: option, analgesia, quelled, maeshowe, comers, inevitably, kazan, burglary,
Nearest to in: embittered, specified, deicide, pontiff, omitted, edifice, levitt, cordell,
Nearest to world: intelligible, unguarded, pretext, cinematic, druidic, agm, embarks,
cingular,
Nearest to use: hab, tabula, estates, laminated, battle, loyola, arcadia, discography,
Nearest to from: normans, zawahiri, harrowing, fein, rada, incorrect, spandau, insolvency,
Nearest to people: diligent, tum, cour, komondor, lecter, sadly, barnard, ebony,
Nearest to it: fulfilled, referencing, paullus, inhibited, myra, glu, perpetuation,
theologiae,
Nearest to united: frowned, turkey, profusion, personifications, michelangelo,
sisters, okeh, claypool,
Nearest to new: infanta, fen, mizrahi, service, monrovia, mosley, taxonomy, year,
Nearest to seven: tilsit, prefect, phyla, varied, reformists, bc, berthe, acceptance,
Nearest to also: pri, navarrese, abandonware, env, plantinga, radiosity, oops, manna,
Nearest to about: lorica, nchen, closing, interpret, smuggler, viceroyalty, barsoom, caving,
Nearest to his: introduction, mania, rotates, switzer, elvis, warped, chilli,
etymological,
Nearest to and: robson, fun, paused, scent, clouds, insulation, boyfriend, agreeable,
Average loss at step  2000 :  113.878970229
Average loss at step  4000 :  53.0354625027
Average loss at step  6000 :  33.5644974816
Average loss at step  8000 :  23.246792558
Average loss at step  10000 :  17.7630081813
```

（5）用t-SNE降维呈现。这里我们将上一步训练的结果做了一个t-SNE降维处理，最终用Matplotlib绘制出图形，图形见图3-19。代码如下：

```
def plot_with_labels(low_dim_embs, labels, filename='tsne.png'):
  assert low_dim_embs.shape[0] >= len(labels), "More labels than embeddings"
  plt.figure(figsize=(18, 18))  # in inches
  for i, label in enumerate(labels):
    x, y = low_dim_embs[i, :]
    plt.scatter(x, y)
    plt.annotate(label,
                 xy=(x, y),
                 xytext=(5, 2),
                 textcoords='offset points',
                 ha='right',
                 va='bottom')

  plt.savefig(filename)

try:
  from sklearn.manifold import TSNE
  import matplotlib.pyplot as plt

  tsne = TSNE(perplexity=30, n_components=2, init='pca', n_iter=5000)
  plot_only = 500
  low_dim_embs = tsne.fit_transform(final_embeddings[:plot_only, :])
  labels = [reverse_dictionary[i] for i in xrange(plot_only)]
  plot_with_labels(low_dim_embs, labels)

except ImportError:
  print("Please install sklearn, matplotlib, and scipy to visualize embeddings.")
```

小知识

t-SNE是流形学习（manifold Learning）方法的一种。它假设数据是均匀采样于一个高维空间的低维流形，流形学习就是找到高维空间中的低维流形，并求出相应的嵌入映射，以实现维数约简或者数据可视化。流形学习方法分为线性的和非线性的两种。线性的流形学习方法如主成份分析（PCA） 非线性的流形学习方法如等距特征映射（Isomap）、拉普拉斯特征映射（Laplacian eigenmaps，LE）、局部线性嵌入（Locally-linear embedding，LLE）等。

3.3.2 嵌入投影仪

在3.2节中我们说到，在TensorBorad的面板中还有一个EMBEDDINGS面板，用于交互式可视化和分析高维数据。对于上面的word2vec_basic.py文件，我们只是做了一个降维分析，下面我们就来看看TensorBorad在词嵌入中的投影。这里采用官方GitHub开源实现上的例子[5]进行讲解。

这里我们自定义了两个操作（operator，OP）：SkipgramWord2vec和NegTrainWord2vec。为什么需要自定义操作以及如何定义一个操作将在4.10节介绍。操作需要先编译，然后执行。这里采用Mac OS系统，在g++命令后加上-undefined dynamic_lookup参数：

```
TF_INC=$(python -c 'import tensorflow as tf; print(tf.sysconfig.get_include())')
g++ -std=c++11 -shared word2vec_ops.cc word2vec_kernels.cc -o word2vec_ops.so -fPIC -I $TF_INC -O2 -D_GLIBCXX_USE_CXX11_ABI=0
```

在当前目录下生成word2vec_ops.so文件，然后执行word2vec_optimized.py，生成的模型和日志文件位于/tmp/，我们执行：

```
tensorboard --logdir=/tmp/
```

访问http://192.168.0.101:6006/，得到的EMBEDDINGS面板如图3-22所示。

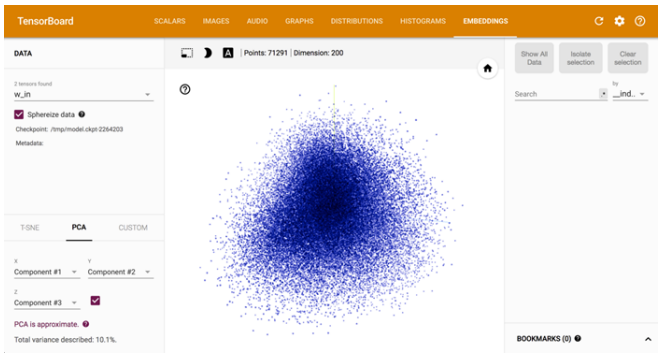

图3-22

在EMBEDDINGS面板左侧的工具栏中，可以选择降维的方式，有T-SNE、PCA和CUSTOM的降维方式，并且可以做二维/三维的图像切换。例如，切换到t-SNE降维工具，可以手动调整Dimension（困惑度）、Learning rate（学习率）等参数，最终生成10 000个点的分布，如图3-23所示。

图3-23

在EMBEDDINGS面板的右侧，可以采用正则表达式匹配出某些词，直观地看到词之间的余弦距离或欧式距离的关系，如图3-24所示

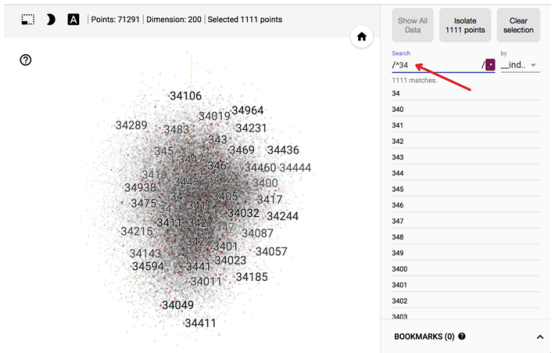

图3-24

任意选择一个点，如8129，选择“isolate 101 points”按钮，将会展示出100个在空间上最接近被选择点的词，也可以调整展示的词的数量，如图3-25所示。

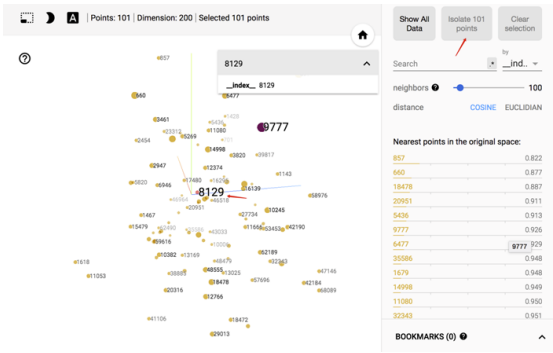

图3-25

3.4 小结

可视化是研究深度学习的一个重要方向，有利于我们直观地探究训练过程中的每一步发生的变化。TensorFlow提供了强大的工具TensorBoard，不仅有完善的API接口，而且提供的面板也非常丰富。在4.3.2节我们会讲解实现TensorBoard的API。在第17章我们还会讲到TensorFlow的调试工具，调试和可视化配合起来，有利于精准地调整模型。